ASSEMBLY SEQUENCE PLANNING

NAGW-1333

By:

A.C. Sanderson
H. Zhang
L.S. Homen de Mello

Department of Electrical, Computer and Systems Engineering
Department of Mechanical Engineering, Aeronautical Engineering & Mechanics
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

August 1989

CIRSSE Document #36

ASSEMBLY SEQUENCE PLANNING

Arthur C. Sanderson and Hui Zhang

Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute,
Troy, NY 12180

Luiz S. Homem de Mello

Jet Propulsion Laboratories
California Institute of Technology
Pasadena, CA 91109

Manuscript prepared for AI Magazine: Special Issue on Assembly Planning

August 26, 1989

ASSEMBLY SEQUENCE PLANNING

Arthur C. Sanderson and Hui Zhang
Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, NY 12180

and

Luiz S. Homem de Mello
Jet Propulsion Laboratories
California Institute of Technology
Pasadena, CA 91109

ABSTRACT

The sequence of mating operations which can be carried out to assemble a group of parts is constrained by the geometric and mechanical properties of the parts, their assembled configuration, and the stability of the resulting subassemblies. An approach to representation and reasoning about these sequences is described here, and leads to several alternative explicit and implicit plan representations. The PLEIDEAS system is intended to provide an interactive software environment for designers to evaluate alternative systems and product designs through their impact on the feasibility and complexity of the resulting assembly sequences.

1. Introduction

Assembly plays a fundamental role in the manufacturing of most products. Parts which have been individually formed or machined to meet designed specifications are assembled together into a configuration which achieves the functions of the final product or mechanism. The economic importance of assembly as a manufacturing process has led to extensive efforts to improve the efficiency and cost effectiveness of assembly operations. In recent years, the use of programmable and flexible automation has enabled the partial or complete automation of assembly of products in smaller volumes and with more rapid product changeover and model transition. Artificial Intelligence plays an increasingly key role in such flexible automation systems. Artificial Intelligence tools facilitate reasoning about geometry, mechanics, and operations for assembly sequence planning.

In practice, manual labor, fixed automation, and flexible automation are often combined in modern manufacturing systems in order to take advantage of cost and reliability trade-offs. Decisions regarding alternative assembly manufacturing technologies, tools for assembly manufacturing system design, and methods for assembly system implementation are key challenges which currently face production engineers. More systematic approaches to the analysis, design, and planning of these assembly systems are needed in order to enhance their performance and to enable their cost-effective implementation. The work described in this paper focuses on the representation of assembly sequence plans and the development of assembly sequence planning tools which form the basis for automated and interactive assembly systems design methods. The PLEIDEAS system, or PLanning Environment for Integrated Design of Assembly Systems, described here, is an effort to develop such a set of software tools.

While assembly has important applications in manufacturing, the assembly process itself has attracted scientific interest as an example of intelligent robotic manipulation. The mating of two parts with complex geometries typically requires the integration of sensory and motor control information with an internal representation of parts, geometries and relationships. Humans carry out these manipulation tasks using well-practiced skills of integration of motor control and sensory interpretation with stored models of geometry. The replication of these skills in automated robotic systems has proven to be extremely difficult. Fundamental issues in robotics, control theory, pattern recognition, and artificial intelligence are posed by this complex task. A number of generic assembly problems in manipulation – "put the peg in the hole" – , sensing – "find the part in the bin"– , and planning – "put block A on block B"— have evolved as classical challenges in the scientific literature and means to evaluate and compare approaches and algorithms. Assembly sequence planning may also be thought of as such a generic scientific problem in which the fundamental properties of assembly relations, geometries, and operations are used to guide the search for correct, complete, optimal, or desirable sequences.

Blocks world may be thought of as a simple assembly planning environment. A goal state in blocks world specifies the contacts between a set of parts, and the PUTON(A,B) operation may be thought of as a mating operation which requires geometric access to the mating surface as well as stability of the resulting configuration. However, many of the domain independent approaches [1–5] to planning in blocks world do not map well into the more generalized assembly problem. The use of a propositional representation for states and subgoals, such as in STRIPS [1], has clear limitations when faced with the more generalized geometries and mechanisms incorporated in product assemblies. An alternative approach to decomposition of the planning problem is needed. While the representation of state is more complex than in blocks world, there are domain-specific ordering constraints for assembly which may be used to simplify the representation of plans. In this paper we will describe the use of the AND/OR graph, directed graph, and precedence relations for assembly sequence plan representation.

Assembly sequence planning is part of a hierarchy of steps in assembly system design for manual, fixed automation, or programmable automation systems. One such hierarchy of design and implementation for a programmable assembly system is shown in Figure 1. In this view,

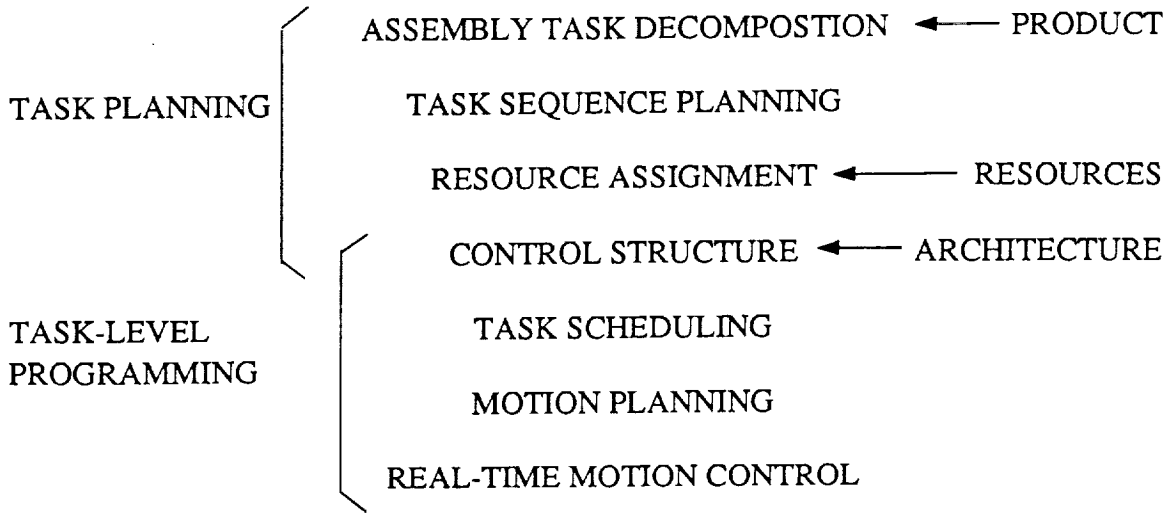

Figure 1 Hierarchy of planning processes for assembly system design and implementation.

task level planning of the system is carried out based on a description of the product and its parts, a description of the system and its resources such as robots, grippers, fixtures, or sensors, and a description of the coordination architecture including computing and communications capabilities. The resulting task-level plan describes the decomposition of the assembly task, the assignment of assembly sub-tasks to system resources, and a model for the coordination and scheduling of systems resources based on architectural features. The implementation of the task level plan is carried out using a task-level programming approach in which the detailed control of paths and trajectories, fine motion, grasping, sensing, and interaction forces, is specified. Real time execution utilizes this planning structure as a framework to provide efficient and reliable performance. Developing such a planning and control framework which can successfully cope with the uncertainties in design specification and execution parameters is a major goal in assembly planning research.

This paper is concerned with planning sequences of assembly operations which satisfy conditions of feasibility and yield states which are stable. The paper provides an overview of an approach and examples of results. Additional details may be found in the previous publications [6–14]. Section 2 describes the relational graph structure which we utilize as the basis for assembly representation, and abstracts the problem of task decomposition to identifying the cut sets of the graph of connections of the product. Section 3 describes alternative assembly sequence representations, and illustrates some advantages and properties of the AND/OR graph. Section 4 defines two types of precedence relations which may be generated from the AND/OR graph. These precedence relations may be simplified when independence properties of the assemblies hold, and, in addition, the real-time properties of precedence relations may be guaranteed under certain conditions. Section 5 discusses the issue of evaluation functions for assembly sequence plans. Section 6 summarizes the PLEIDEAS system which is an approach to developing an environment for assembly system planning. Section 7 presents conclusions and directions of continuing work.

2. Assembly Sequence Planning.

An assembly product description consists of the geometric description of each of the individual parts, their geometric tolerances, and the configuration in which the parts fit together to form the final product. In practice, such a description is often incomplete or inexact, and relies strongly on human experience and intuition for the full interpretation of this description. For purposes here, we will assume that an explicit description of the parts geometry and the assembled configurations is given or derived. Our approach to assembly representation is strongly influenced by previous experience with relational models [15,16,17]. We do not currently consider tolerances for planning purposes, although the geometric modeling system which we use incorporates them (see Section 6) and provides access for use in the planning procedures. In addition to the geometric configuration, most assemblies utilize attachments which apply physical forces to constrain the relative motion among parts and stabilize the final assembly. While attachments such as screws and clips are represented geometrically in the assembly description, it is often impossible to infer from their geometric description what the physical role of these parts may be as an attachment for the assembly. We therefore add the attachment description interactively to the relational model.

In our work, three levels of description are utilized for assembly product representation. An example for a simple product is shown in Figure 2. (1). Figure 2a illustrates a complete CAD description of individual parts geometries as output by the CATIA (IBM Corp.) system. (2). From the CAD-based description of parts, we derive a relational graph model of the resulting assembly as shown in Figure 2b. This relational graph extraction is based on an object-oriented geometric modeling system, GEOS (see Section 6), which provides an explicit representation of parts and contacting surfaces in the assembly. We interactively add attachment entities in order to explicitly describe constraints on the degrees of freedom of contacts within the structure. In this description each attachment has an agent and a contact, where each contact has its degrees of freedom constrained when the attachment agent is present. These relationships directly influence the feasible assembly sequences since an attachment agent must often be removed prior to removal of a part which breaks a contact. (3). A simplified relational structure called the graph of connections is shown in Figure 2c. While the relational graph contains specific properties and attributes associated with each of the entities, the graph of connections is purely the relational structure which defines the parts and connections. The three levels of product description shown in Figure 2 are all necessary for assembly sequence planning, and this hierarchical organization facilitates the planning tasks.

There are ten different feasible sequences which may be used to assemble the product shown in Figure 2. These sequences are listed in Figure 3. From a planning perspective, each of these sequences is a series of actions which satisfies the preconditions for each action and which leads to the goal state. In assembly sequence planning, the initial state is always that state in which no parts are interconnected and the goal state is always that in which all parts are interconnected in the final unique configuration. The intermediate states of the system consists of subsets of mutually interconnected parts called sub-assemblies. We will assume for this discussion that the

geometric configuration of a sub-assembly is uniquely specified by its constituent parts, that is, there is only one way for a given subset of parts to fit together. For these examples therefore, if we assemble a cylinder with a piston, we would uniquely specify the final position of the piston and assume that it would not have to occupy multiple discrete states in order to enable successful assembly.

Assembly Task Feasibility Predicates

An assembly action mates two parts or sub-assemblies to form a new sub-assembly. (For the current discussion we will assume that only two, and not more, sub-assemblies are joined at a given time.) In assembly, the preconditions for such an action may be thought of as the feasibility conditions for the execution of the operation and the acceptability of the resulting state. Preconditions on the feasibility of operations may be broken down into several different categories:

1. Resource independent feasibility conditions: Independent of the specific robots, grippers, or fixtures which are utilized in the assembly, there are geometric constraints among the parts which restrict the order in which operations can be carried out. These restrictions include: a) *Local geometric feasibility* - Is local or incremental translation of the part feasible from its final assembled state? This question may be answered for translational motion using only surface normal information which is stored in the relational graph. See [13] for details. b) *Global geometric feasibility* - Is there an unobstructed path from some remote position to the final assembled state given the geometric obstructions posed by other parts which are present in the assembly state? This problem is related to the general path planning problem [18], but only requires determination of the existence of a path, and not necessarily the specification of the path.

2. Resource dependent feasibility conditions: a) *Geometric feasibility* - Is there an unobstructed path for mating of parts given the geometry of the parts plus the geometry of robots, grippers, and fixtures which may be utilized in the operation? B) *Attachment feasibility* - Can appropriate forces be exerted by available tools in order to attach parts as required by the mating operation? c) *Tool availability* - Is there an appropriate robot, gripper, or fixture available in order to carry out a particular task to meet geometric and attachment constraints?

While such feasibility conditions might be examined simultaneously, it is often more efficient to structure the planning approach hierarchically. As suggested by Figure 1, we can examine assembly sequence plans which are feasible from a resource independent point of view, and these plans subsume all of the feasible plans which incorporate resource dependencies. The resource independent plan reduces the search space for the resource dependent planning. In addition, a number of heuristics related to the complexity of operations and the desirability of states may be added at this stage to further reduce the search space. Figure 3 shows a set of assembly sequences for the simplified flashlight product which are all feasible from a resource independent criterion. An example of an infeasible sequence for this product would be sequence A–C–B. In this sequence the cap and handle are attached to the receptacle before the stick is placed inside. It becomes geometrically infeasible to insert the stick into the receptacle when both ends are attached. Such a sequence should be rejected early in the planning process and

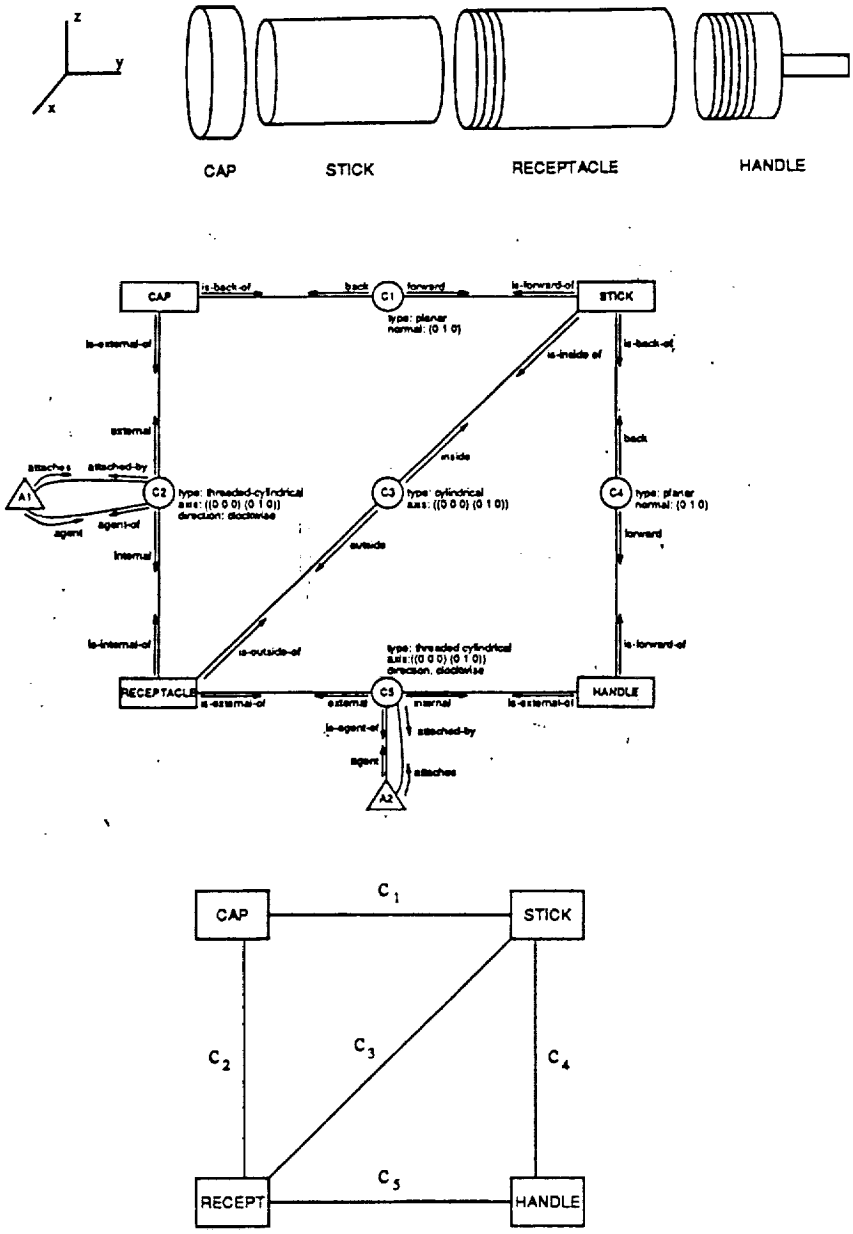

Figure 2 Three levels of representation of assembly product: (a) Solid model of parts, (b) Relational model of assembly, (c) Graph of connections of assembly.

not considered further. Another sequence such as B–A–C might be geometrically feasible, but undesirable. In this case, the stick is inserted into the receptacle resulting in a sub-assembly state which may require fixturing in order to keep the stick from sliding out. Such a state might be detected and evaluated as undesirable by an appropriate heuristic. Resource dependent feasibility

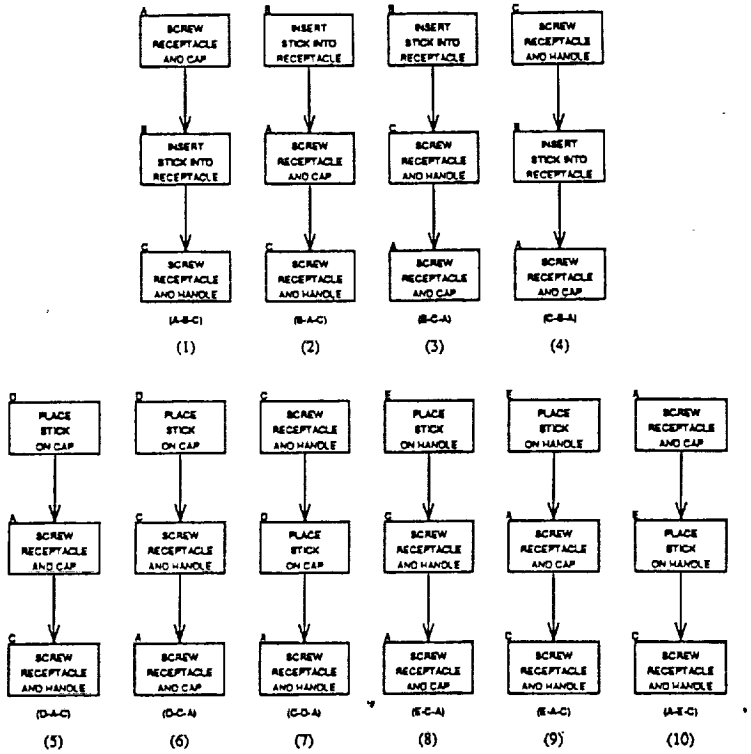

Figure 3 Feasible assembly sequences for the product shown in Figure 2.

constraints could further limit the set of sequences shown in Figure 3. If there were no gripper available to pick up the stick or no mechanism available to screw in the handle then there might be no remaining feasible sequences.

In addition to task feasibility conditions, we need to consider state feasibility conditions. Again, several resource independent and resource dependent conditions may come into play. These include:

1. *Stability* - Is there one pose of the sub-assembly for which the configuration is stable? This is a very difficult problem in general [19], is strongly influenced by the types of friction and surface interaction assumptions which are made.
2. *Rigidity* - Is the sub-assembly stable for all poses of the sub-assembly?

Each of these criteria may be examined under different assumptions:

1. Resource independent constraints: a) No gravity or external forces - in this case all sub-assemblies are stable and rigid. b) Gravity or other external forces - in this case stability may strongly depend upon orientation of the sub-assembly, and rigidity often requires attachment of all parts.
2. Resource dependent constraints: Fixtures or grippers may be added to the configuration in order to impose stability or rigidity on a sub-assembly during the assembly process.

The assembly sequences presented in Figure 3 all have at least one stable pose for each state in the presence of gravity, although the pose might be changed during the sequence to insure stability and also enable geometric feasibility. In general, the assessment of the stability of a state would also require a description of the frictional forces which arise.

Assembly Sequence Planning

In our work, we provide a framework for assembly sequence planning in which many different feasibility criteria might be incorporated. The same approach to plan generation, plan representation, and plan evaluation may be used for different feasibility conditions. As discussed above, the resource independent plans subsume the resource dependent plans and therefore simplify the overall planning process. In the experiments described here, we have implemented two feasibility criteria for the generation of assembly sequences: 1) Resource independent local geometric feasibility, and 2) Attachment feasibility based on the assumption that attachment tools are available, and that breaking attached contacts is infeasible unless the attachment agent is removed. These two feasibility criteria are sufficiently interesting and complex to illustrate many of the important problems which occur in assembly sequence planning.

The problem of generating feasible assembly sequences for a product may be transformed into the problem of generating disassembly sequences for the same product. For many problems of interest this transformation reduces the branching factor of the search space since there are often many more options and dead ends which occur in assembly than in disassembly. This transformation of the problem is conceptual rather than physical. Assembly tasks are not necessarily reversible in the physical sense, but we impose this reversibility from a conceptual standpoint in order to plan correct sequences. This decomposition approach to assembly planning maps nicely onto our relational model of the product, and leads to a recursive decomposition of the task in which each decomposable subproblem is a proper subset of the original model. In addition, as we will see in the next section, this decomposition approach lends itself to the AND/OR graph representation of assembly sequences.

The basic planning strategy is to recursively enumerate the decompositions of the assembly and to select those decompositions that are feasible by imposing a set of feasibility predicates. The decompositions of the assembly are enumerated by the cut sets of the assembly's graph of connections. The feasible operations and feasible states predicates are evaluated using the relational model and the geometric parts models.

Assembly States

The assembly graph of connections can be represented by a simple undirected graph $\langle P, C \rangle$ in which the parts $P = \{p_1, p_2, \cdots, p_N\}$ are the set of nodes, and the connections $C = \{c_1, c_2, \cdots, c_L\}$ are the set of edges. A state of the assembly process is a configuration of the parts at the beginning or at the end of an assembly task. The state of an assembly process may be characterized either by the configuration of contacts which have been established or by the partitions of the parts which are connected. In the first case, a state of the assembly process may be represented by an L-dimensional binary vector $\mathbf{x} = [x_1, x_2, \cdots, x_L]$ in which the i^{th} component of x_i is true or false respectively if the i^{th} connection is established in that state or

not. For example, if the first task of the assembly process for the example shown in Figure 2 is the joining of the cap to the receptacle, the second state of the assembly process can be represented by [false, true, false, false, false].

Alternatively, an assembly state may be characterized by partitioning of parts into sub-assemblies. For example, if the first task is the joining of the cap to the receptacle, the second state of the assembly process can be represented by { {CAP, RECEPTACLE}, {STICK}, {HANDLE} }. Given an assembly's graph of connections and one of the two representations of the assembly state, it is straightforward to obtain the other representation. It is also important to observe that not all partitions and not all L-dimensional binary vectors can characterize a proper state of the assembly. For example, for the assembly shown in Figure 2, the 5-dimensional binary vector [true, true, false, false, false] does not correspond to a state because if connections c_1 and c_2 are established then c_3 must also be established. A sub-assembly predicate *sa* is defined to determine whether a subset of parts makes up a sub-assembly and, therefore, whether a given binary vector is, in fact, a proper state vector of the assembly. In addition, we define a sub-assembly's stability predicate *st* which determines whether a sub-assembly described by its set of parts and connections is stable. While we have studied a sub-assembly's stability predicate based on the stability of parts configuration subject to gravity and zero friction, many other assumptions and analysis tools or heuristics might be implemented here. An assembly state representation in which all sub-assemblies satisfy the stability predicate is said to be a stable assembly state representation.

Assembly Tasks

Given two sub-assemblies characterized by their sets of parts θ_i and θ_j, we say that joining θ_i and θ_j is an assembly task if the set $\theta_k = \theta_i \cup \theta_j$ characterizes a sub-assembly. Equivalently, an assembly task can be characterized by the output sub-assembly and the set of connections that are established by the task. In order for such a set of connections to represent an assembly task, it must correspond to a cut set of the graph of connections of the task's output sub-assembly. Conversely, each cut set of a sub-assembly's graph of connections corresponds to an assembly task.

Each assembly task must be evaluated by the operations feasibility predicates described above. An assembly task is said to be resource independent geometrically feasible if there is a collision-free path to bring the two sub-assemblies into contact from a situation in which they are far apart. We further decompose this geometric feasibility predicate *gf* into a local predicate and a global predicate. The local predicate tests for the incremental motion of the designated sub-assemblies, while the global predicate tests for the existence of a global collision-free path. An assembly task is said to have attachment feasibility if it is feasible to establish the attachments that act on the contacts between the two sub-assemblies. The attachment feasibility predicate *af* evaluates this predicate based on the attachment description which is incorporated into the relational model. A local geometric feasibility can be evaluated also using the contact attributes and properties established in the relational model. Global geometric feasibility requires access to the detailed parts descriptions which are referenced from the relational model to the parts models.

Again, the computation of these predicates may be approached in a variety of different ways. They are extremely complex to compute in general, and incorporate many very challenging and interesting problems in geometric and physical reasoning. As mentioned above, the experiments described in this paper include a local geometric feasibility predicate and an attachment feasibility predicate, in order to demonstrate the overall planning framework.

Our algorithm for the generation of assembly sequences utilizes an approach which enumerates the decompositions of the assembly and then selects those decompositions that are feasible. As shown in the next section, this recursive decomposition results in the construction of the AND/OR graph representation of assembly plans. Figure 4 outlines the procedure GET-FEASIBLE-DECOMPOSITIONS which takes as input the relational model of an assembly and returns all feasible decompositions of that assembly. The procedure first generates the graph of connections for the input assembly and computes the cut sets of this graph. The cut sets are enumerated by looking at all connected subgraphs having the cardinality of their set of nodes smaller than or equal to half of the cardinality of the set of nodes in the whole graph. For each of these subgraphs, the set of edges of the whole graph that have only one end in the subgraph defines a cut set, if their removal leaves the whole graph with exactly two components. Each resulting cut set corresponds to a decomposition of the graph. The procedure GET-DECOMPOSITION finds that decomposition, and the procedure FEASIBILITY-TEST is used to check whether that decomposition is feasible or not. Note that the procedure FEASIBILITY-TEST further breaks down into a set of procedures which implements the individual operations feasibility predicates and state feasibility predicates.

3. Assembly Sequence Representation

As described above, each assembly may have many different feasible assembly sequences. Our first objective is to generate those sequences and represent them in an efficient manner. Given an assembly that has N parts, an ordered set of N-1 assembly tasks $\tau_1\ \tau_2\ \cdots\ \tau_{N-1}$ is an assembly sequence if there are no two tasks that have a common input sub-assembly, the output sub-assembly of the last task is the whole assembly, and the input sub-assemblies to any task is either a one-part sub-assembly or the output sub-assembly of a task that precedes. Such an assembly sequence can also be characterized by an ordered sequence of states, in which the state s_1 is the state in which all parts are separated, the state s_N is the state in which all parts are joined forming the whole assembly, any two consecutive states are such that only the two input sub-assemblies of the task are in s_i and not in s_{i+1}, and only the output sub-assembly of task τ_i is in s_{i+1} and not in i. An assembly sequence is said to be feasible if all its assembly tasks are feasible, and all its assembly states are feasible.

An assembly sequence therefore can be represented in several different ways: 1) An ordered list of task representations. 2) An ordered list of binary vectors. 3) An ordered list of partitions of the set of parts. 4) An ordered list of subsets of connections. For example, a feasible assembly sequence for the product shown in Figure 2 could be represented as:

```
procedure GET-FEASIBLE-DECOMPOSITIONS(assembly)
feasible-decompositions ← NIL
graph ← GET-GRAPH-OF-CONNECTIONS(assembly)
cut-sets ← GET-CUT-SETS(graph)
while cut-sets is not empty do
    begin loop1
    next-cut-set ← FIRST(cut-sets)
    cut-sets ← TAIL(cut-sets)
    next-decomposition ← GET-DECOMPOSITION(next-cut-set)
    if FEASIBILITY-TEST(next-decomposition)
        then feasible-decompositions ← UNION(feasible-decompositions, LIST(next-decomposition))
    end loop1
return feasible-decompositions
end procedure
```

Figure 4 Procedure GET-FEASIBLE-DECOMPOSITIONS generates the feasible decompositions of the assembly.

- A three-element list of task representations:

```
({{CAP},{RECEPTACLE}}
 {{CAP,RECEPTACLE},{STICK}}
 {{CAP, RECEPTACLE, STICK},{HANDLE}})
```

- A four-element list of 5–dimensional binary vectors:

```
([false,false,false,false,false]
 [false,true,false,false,false]
 [true,true,true,false,false]
 [true,true,true,true,true])
```

- A four-element list of partitions of the set of parts:

```
({{CAP},{RECEPTACLE},{STICK},{HANDLE}}
 {{CAP,RECEPTACLE},{STICK},{HANDLE}}
 {{CAP,RECEPTACLE,STICK},{HANDLE}}
 {{CAP,RECEPTACLE,STICK,HANDLE}})
```

- A three-element list of sets of connections: $(\{c_2\}\{c_1,c_3\}\{c_4,c_5\})$.

Since each assembly sequence can be represented by ordered lists, it is possible to represent the set of all assembly sequences by a set of lists. While this set of lists may represent a complete and correct description of all feasible assembly sequences, it is not necessarily the most compact or most useful representation of those sequences. In particular, since many assembly sequences share common subsequences, and common states, attempts have been made to create more compact representations that can encompass all feasible assembly sequences. We have utilized the AND/OR graph representation of assembly sequences as a basis for these representations. We have developed an algorithm for the generation of the AND/OR graph and subsequent algorithms which generate equivalent, complete and correct representations of the directed graph and precedence relations.

The nodes in this AND/OR graph representation of assembly sequences are the subsets of P that characterize stable sub-assemblies. The hyperarcs correspond to the feasible assembly tasks. Each hyperarc is an ordered pair in which the first element is a node that corresponds to a stable sub-assembly θ_k, the second element is a set of two nodes, $\{\theta_i,\theta_j\}$ such that $\theta_k = \theta_i \cup \theta_j$, and the assembly task characterized by θ_i and θ_j is feasible. Each hyperarc is associated with a decomposition of the sub-assembly that corresponds to its first element and can also be characterized by this sub-assembly and the subset of all its connections that are not in the graphs of connections of the sub-assemblies in the hyperarc's second element. This subset of connections associated to a hyperarc corresponds to a cutset in the graph of connections of the sub-assembly in the hyperarc's first element. This AND/OR graph can be formally defined as follows:

Definition: The AND/OR graph of feasible assembly sequences of an assembly whose set of parts is $P = \{p_1, p_2, \cdots, p_N\}$ is the AND/OR graph $\langle S_P, D_P \rangle$ in which

$$S_P = \{\theta \in \Pi(P) \mid sa(\theta) \wedge st(\theta)\}$$

is the set of stable subassemblies, and

$$D_P = \{(\theta_k, \{\theta_i, \theta_j\}) \mid [\theta_i, \theta_j, \theta_k \in S_P] \wedge [U(\{\theta_i, \theta_j\}) = \theta_k] \wedge [af(\{\theta_i, \theta_j\})] \wedge [gf(\{\theta_i, \theta_j\})]\}$$

is the set of feasible assembly tasks. The notation $\Pi(P)$ is used to represent the set of all subsets of P

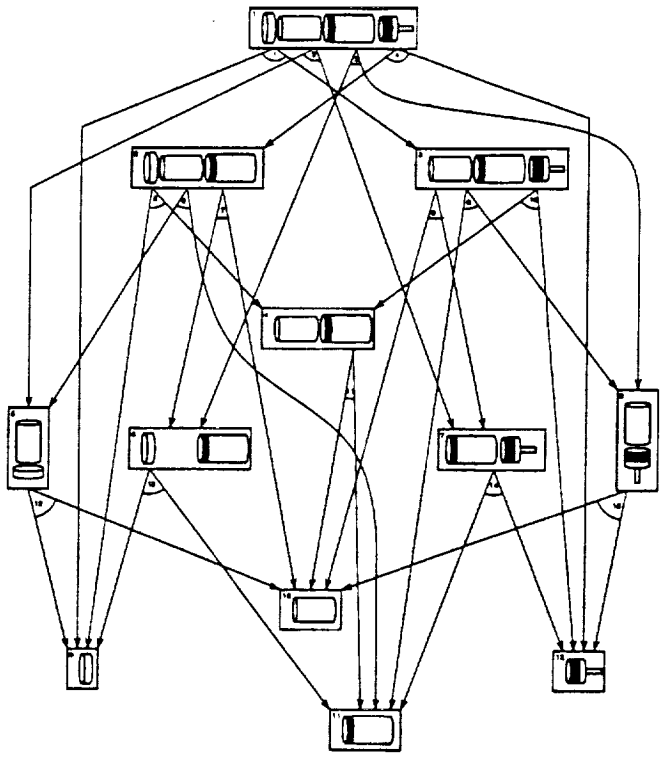

Figure 5 AND/OR graph of the product shown in Figure 2.

As an example, Figure 5 shows the AND/OR graph for the feasible sequences for the assembly shown in Figure 2. This AND/OR graph representation is complete and correct in that it includes all possible feasible assembly sequences, and also does not include any infeasible assembly sequences. A given assembly sequence can be defined as a feasible assembly tree within the AND/OR graph.

The algorithm GENERATE–AND/OR–GRAPH takes the relational model of an assembly and returns the AND/OR graph representation of all assembly sequences for that assembly. The nodes in the AND/OR graph returned are pointers to relational models of sub-assemblies. The algorithm is not reproduced here due to space limitations, but it uses procedure GET–DECOMPOSITIONS to generate all decompositions of the relational models and keeps track of lists of pointers to determine whether specific sub-assemblies have previously been generated or not. These procedures may further be made more efficient by linking together the evaluation of feasibility tests between different decompositions to avoid duplication of the computational work. An analysis of the completeness, correctness and complexity of the AND/OR graph generation algorithm is discussed in [12]

Given an assembly whose graph of connections is $\langle P, C \rangle$, a directed graph can also be used to represent the set of all feasible assembly sequences. The nodes in this directed graph correspond to stable state partitions of the set P. These are the partitions Θ of P such that if $\theta \in \Theta$ then θ is a stable sub-assembly of P. The edges in this directed graph are ordered pairs of nodes. For any edge, there are only two subsets, θ_i and θ_j, in the state partition corresponding to the first node that are not in the state node corresponding to the second node. Therefore, each edge corresponds to an assembly task. If all assembly tasks are feasible then the graph is referred to as the directed graph of feasible assembly sequences. Figure 7 shows the directed graph of feasible assembly sequences for the assembly shown in Figure 2. A path in the directed graph of feasible

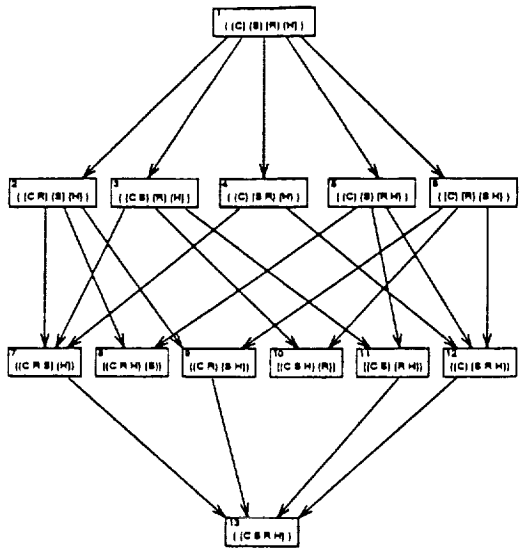

Figure 6 Directed graph of the product shown in Figure 2.

assembly sequences corresponds to a feasible assembly sequence for the assembly P. In such a path, the ordered sequence of edges corresponds to the ordered sequence of tasks, while the ordered sequence of nodes corresponds to the ordered sequence of states of the assembly process. The equivalence of the directed graph and the AND/OR graph as complete and correct representations of feasible assembly sequences has been proven elsewhere [12]. The relative complexity of the two representations in terms of the number of nodes in the graphs has also been addressed in [12]. In that analysis, for strongly connected assemblies, the AND/OR graph had fewer nodes for all assemblies larger than $N = 4$, and at $N = 10$ it had fewer nodes by more than two orders of magnitude. While the list of sequences, the directed graph, and the AND/OR graph all represent complete sets of sequences, the AND/OR graph has advantages in the efficiency of its representation. This efficiency may be viewed as a space/time tradeoff, such that, given a fixed amount of space allocated for representation storage, the AND/OR graph will tend to represent more feasible sequences and therefore may lead to more optimal performance, shorter time, solution. An example of the use of the AND/OR graph representation for the opportunistic scheduling of robotic assembly tasks is shown in [12].

4. Precedence Relations

It has been observed that many assembly problems have inherent ordering constraints which dominate the selection of feasible sequences for assembly systems. For the example in Figure 2, "The stick must be in the receptacle before both ends are attached" is a generalization on ordering which, in itself, is sufficient to distinguish all the feasible and infeasible sequences. Intuitive precedence relations of this type have been used by assembly designers for many years. The work of Bourjault [20] and DeFazio and Whitney [21] and Lui [22] has attempted to capture

this intuitive knowledge by formalized sets of interactive questions provided to the designer. In our work, we have shown how to derive these sets of precedence relations directly from the AND/OR graph, and therefore to be able to generate precedence relations automatically from the design description.

In this section, we define two types of precedence relations: Type 1 - precedence relations between the establishment of one connection and states of the assembly process, and Type 2 - precedence relations between the establishment of one connection and the establishment of another connection of the assembly process. Both of these types of precedence relations result in logical expressions on the occurrence of connections or states. They both may be generated directly from the AND/OR graph and shown to be complete and correct descriptions of feasible assembly sequences. We also introduce two independence properties of assemblies which permit the simplification of these precedence relations. These independence properties in themselves provide some interesting insight into the characteristics of assemblies and the complexity of their resulting sequences.

Type 1 Precedence Relations: Connection-State

If we represent the states of the assembly process by L-dimensional binary vectors, then a set of logical expressions can be used to encode the directed graph of feasible assembly sequences. Let $\Xi_i = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_3\}$ be the set of states from which the i^{th} connection can be established without precluding the completion of the assembly. The establishment condition (Bourjault, [20]) for the i^{th} connection is the logical function:

$$F_i(\mathbf{x}) = F_i(x_1, x_2, \cdots, x_L) = \sum_{k=1}^{K_i} \prod_{l=1}^{L} \gamma_{kl},$$

where the sum and the product are the logical operations OR and AND respectively, and γ_{kl} is either the symbol x_l if the l^{th} component of $\boldsymbol{x_k}$ is true, or the negation $\overline{x_l}$ if the l^{th} component of $\boldsymbol{x_k}$ is false. Clearly, every element $\boldsymbol{x_k}$ of $\Xi_i = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_3\}$ is such that $F_i(\boldsymbol{x_k})$ = true. It is often possible to simplify the expression of $F_i(\boldsymbol{x_k})$ using the rules of Boolean algebra. The set of establishment conditions is a correct and complete assembly sequences, and this set of conditions can be obtained directly from the AND/OR graph as described in [14]. A complementary set of conditions for the infeasible assembly states may be used as a basis for deriving the Type 1 precedence relations.

We will use the notation $C_i \rightarrow S(\boldsymbol{x})$ to indicate that the establishment of the i^{th} connection must precede any state S of the assembly process for which the value of the logical function $S(\boldsymbol{x})$ is true. The argument of $S(\boldsymbol{x})$ of the assembly process for which the value of the logical function $S(\boldsymbol{x})$ is the L-dimensional binary vector representation of the state s. We will use a compact notation for logical combinations of precedence relations. For example, we will write $c_i + c_j \rightarrow S(\mathbf{x})$ when we mean $[c_i \rightarrow S(\mathbf{x})] \vee [c_j \rightarrow S(\mathbf{x})]$. An assembly sequence whose representation as an ordered sequence of binary vectors is $(\boldsymbol{x_1}\ \boldsymbol{x_2}\ \ldots\ \boldsymbol{x_N})$ and whose representation as an ordered sequence of subsets of connections is $(\gamma_1\ \gamma_2\ \ldots\ \gamma_{N-1})$ satisfies the precedence relationship $c_i \rightarrow S(\boldsymbol{x})$.

Let Ψ_S be the set of assembly states that never occur in any feasible assembly sequence. These include the unstable assembly states plus stable states from which the final state cannot be reached plus the states that cannot be reached from the initial state. Let $\Psi_X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_J\}$ be the set of all L-dimensional binary vectors that represent the assembly states in Ψ_S. Every element $\boldsymbol{x}_j$ of Ψ_X is such that the value of the logical function $G(\boldsymbol{x}_j)$ is true where

$$G(\mathbf{x}) = G(x_1, x_2, \cdots, x_L) = \sum_{k=1}^{K} \prod_{l=1}^{L} \lambda_{kl}.$$

The sum and the product in this equation are the logical operations OR and AND, respectively, and λ_{kl} is either the symbol x_l if the l^{th} component of $\boldsymbol{x}_k$ is true, or the symbol $\overline{x_l}$ if the l^{th} component of $\boldsymbol{x}_k$ is false. In many cases the expression $G(\mathbf{x})$ can be simplified using the rules of Boolean algebra. Allowing for simplifications, but keeping the logical function as a sum of products, this equation can be rewritten as

$$G(\mathbf{x}) = \sum_{j=1}^{J'} g_j(\mathbf{x})$$

where each term $g_j(\boldsymbol{x})$ is the product of a subset of $\{x_1, x_2, \cdots, x_L, \overline{x_1}, \overline{x_2}, \cdots, \overline{x_L}\}$ that does not include both x_i and $\overline{x_i}$ for any i. Each term $g_j(\boldsymbol{x})$ can further be rewritten grouping all the nonnegated variables first and all the negated variables last, for example: $g_j(\mathbf{x}) = x_a \cdot x_b \cdot \cdots \cdot x_h \cdot \overline{x_p} \cdot \overline{x_q} \cdot \cdots \cdot \overline{x_z}$.

Any assembly sequence that includes a state that is in Ψ_S is an unfeasible assembly sequence. Therefore a necessary condition for the feasibility of an assembly sequence whose representation as an ordered list of binary vectors is $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ is such that $G(\boldsymbol{x}_1) = G(\boldsymbol{x}_2) = \ldots = G(\boldsymbol{x}_N) =$ *false*.

This condition is equivalent to $g_j(\boldsymbol{x}_i) =$ *false* for $i = 1,2, \ldots, N$ and for $j = 1,2, \ldots, J'$. This necessary condition is also sufficient if the assembly has the following property:

> **Property 1:** Given any two states s_i and s_j, not necessarily in the same assembly sequence, let γ_i and γ_j be the sets of connections that are established in assembly tasks τ_i and τ_j from s_i and s_j respectively. If
>
> $\langle P, C_i \rangle$is the state's graph of connections associated to s_i,
> $\langle P, C_j \rangle$is the state's graph of connections associated to state s_j,
> $\gamma_i \subseteq \gamma_j$,
> $C_i \subseteq C_j$, and
> τ_j is geometrically and mechanically feasible,
> then, τ_i is geometrically and mechanically feasible.

This property corresponds to the fact that if it is feasible to establish a set of connections when many other connections have already been established, then it is also feasible to establish fewer connections when fewer other connections have been established. Although many common

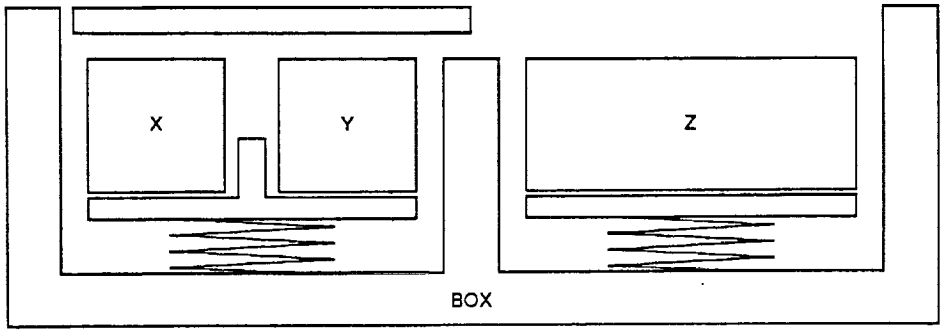

Figure 7 Example of an assembly which violates Property 1. The cover of the box is a sliding mechanism, and the blocks are placed on springs. When either X or Y, but not both, is present, the spring is not completely compressed, the other block prevents the cover from sliding and Z cannot be inserted. Adding the other block to the assembly compresses the spring, permits the cover to slide open, and Z may be inserted. Since insertion of Z is feasible with X and Y present, but not with only X or Y individually present, the assembly violates Property 1.

assemblies have this property, there are assemblies that don't have it. One may view Property 1 as a kind of independence property among connections. It assumes that no mechanisms are present in the assembly such that when additional parts are added the geometric access for assembly of parts is improved. Note that Property 1 does not guarantee that the resulting state will be stable. In the Type 1 precedence relations, the state feasibility is checked explicitly by enumerating unfeasible states, and therefore no prior assumptions are required. For Type 2 precedence relations, an additional independence property is required to guarantee state feasibility of the resulting sequences. An example of an assembly which does not have Property 1 is shown in Figure 8.

In [11,13] we showed that if $(x_1, x_2, \ldots, x_N)$ is an ordered list of binary vectors that represents an assembly sequence, the condition $g_j(x_i) = false$, $i = 1,2, \ldots, N$ is a requirement for a feasible assembly sequence and also corresponds to a precedence relationship:

$$c_p + c_q + \cdots + c_z \rightarrow S(\mathbf{x})$$

where

$$S(\mathbf{x}) = \prod_{l=1}^{L} \lambda_l = \begin{cases} x_l \; if \; l \in \{a, b, \cdots, h\} \\ true \; otherwise \end{cases}.$$

By applying this result to each of the J' terms on the right side of the equation , we obtain J' precedence relationships. Given an assembly sequence, if it satisfies all J' precedence relationships then it does not include any state in Ψ_S and therefore is feasible. Conversely, if the

assembly sequence does not include any state in Ψ_S and therefore it is a feasible assembly sequence then it satisfies all precedence relationships. Therefore the set of J' precedence relationships is a correct and complete representation of the set of all feasible assembly sequences. This result is proven as a theorem in [12,14]. The following example illustrates the application of this result. An algorithm for the generation of these Type 1 precedence relations from the AND/OR graph is given in [14].

As an example of the generation of Type 1 precedence relations, consider the example shown in Figure 2, which has Property 1. In this example, the infeasible states are

$$\Psi_X = \{[false, true, false, false, true], [true, false, false, true, false]\}.$$

Therefore,

$$G(\mathbf{x}) = G(x_1, x_2, x_3.x_4, x_5) = \overline{x_1} \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4} \cdot x_5 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 \cdot \overline{x_5}.$$

The resulting precedence relations are:

$$c_1 + c_3 + c_4 \rightarrow x_2 \cdot x_5,$$
$$c_2 + c_3 + c_5 \rightarrow x_1 \cdot x_4 \quad .$$

A simpler set of precedence relations can be obtained if, in addition, we set non-state vectors as don't care conditions for the simplification. In this case, we obtain the precedence relations:

$$c_1 \rightarrow x_2 \cdot x_5,$$
$$c_2 \rightarrow x_1 \cdot x_4.$$

These sets of expressions are not unique, that is, other logically equivalent sets of precedence relations can be obtained as simplifications of this set. All of these relations express our intuitive understanding of the problem that the stick must be connected prior to closing of the receptacle, and that the stick cannot be joined to the cap and handle without the receptacle already present.

Type 2 Precedence Relations: Connection-Connection

Type 2 precedence relations establish the occurrence of one connection prior to or simultaneously with the occurrence of another connection in the sequence. We use the notation c_i<c_j to indicate that connection c_i must precede connection c_j, and we use the notation $c_i \leq c_j$ to indicate that connection c_i must precede or be simultaneously with the establishment of connection c_j . Furthermore, we use a compact notation for logical combinations of precedence relations: for example we will write $c_i < c_j \cdot c_k$ to mean $(c_i < c_j) \wedge (c_i < c_k)$, and we will write $c_i + c_j < c_k$ to mean $(c_i < c_k) \vee (c_j < c_k)$. An assembly sequence whose representation as an ordered sequence of binary vectors is (x_1,x_2, . . . ,x_N), and whose representation as an ordered sequence of subsets of connections is (γ_1 γ_2 . . . γ_{N-1}) satisfies the precedence relationship c_i<c_j if $c_i \in \gamma_a$, $c_j \in \gamma_b$, and a<b. Similarly, the sequence satisfies $c_i \leq c_j$ if $c_i \in \gamma_a$, $c_j \in \gamma_b$, and $a \leq b$. For example, for the assembly shown in Figure 2, the assembly sequence whose representation as an ordered sequence of binary vectors is

```
([false,false,false,false,false]
 [true,false,false,false,false]
 [true,true,true,false,false]
 [true,true,true,true,true],
```

and whose representation as an ordered sequence of subsets of connections is

$$(\{c_1\}\ \{c_2, c_3\}\ \{c_4, c_5\})$$

satisfies the precedence relationships $c_2 < c_4$ and $c_2 \leq c_3$ but does not satisfy the precedence relationships $c_2 < c_3$ and $c_2 \leq c_1$.

Each feasible assembly sequence of a given assembly can be uniquely characterized by a logical expression consisting of the conjunction of precedence relationships between the establishment of one connection and the establishment of another connection. Given an assembly made up of N parts whose graph of connections is <P,C> let

$$\left\{\left(\gamma_{11}\,\gamma_{21}\,\cdots\gamma_{(N-1)1}\right),\ \left(\gamma_{12}\,\gamma_{22}\,\cdots\,\gamma_{(N-1)2}\right),\ \cdots,\ \left(\gamma_{1M}\,\gamma_{2M}\,\cdots\,\gamma_{(N-1)M}\right)\right\}$$

be a set of M ordered sequences of subsets of connections that represent feasible assembly sequences. Then a correct and complete set of logical expressions representing these sequences is:

$$\prod_{i=1}^{L}\sum_{j=1}^{M}[H_{ij} \leq c_i \leq T_{ij}],$$

where

$$H_{ij} = \prod_{k=1}^{L}\lambda_{ik}\ \ with\ \ \lambda_{ik} = \begin{cases} c_k\ if\ c_k \in \gamma_{ij}\ and\ l \geq i \\ true\ \ otherwise, \end{cases}$$

$$T_{ij} = \prod_{k=1}^{L}\lambda_{ik}\ \ with\ \ \lambda_{ik} = \begin{cases} c_k\ if\ c_k \in \gamma_{ij}\ and\ l \leq i \\ true\ \ otherwise. \end{cases}$$

While it is often possible to simplify this logical expression directly, there is another interesting simplification which arises from the definition of a second independence property of assemblies:

> **Property 2:** Given any three states s_i, s_j, s_k, not necessarily in the same assembly sequence, let $\gamma_i, \gamma_j, \gamma_k$ be the sets of connections that are established in assembly tasks τ_i, τ_j, τ_k from s_i, s_j, s_k respectively. If
>
> $\langle P, C_i\rangle$is the state's graph of connections associated to s_i,
> $\langle P, C_j\rangle$is the state's graph of connections associated to state s_j,
> $\langle P, C_k\rangle$is the state's graph of connections associated to s_k,

$\gamma_i \subseteq \gamma_j \subseteq \gamma_k$,
$C_i \subseteq C_j \subseteq C_k$,
τ_i and τ_k are geometrically and mechanically feasible, and
s_{i+1} and s_{k+1} are feasible states
then, τ_j is geometrically and mechanically feasible, and s_{j+1} is a feasible state.

If Property 2 holds for an assembly, then the following simplified set of Type 2 precedence relations holds:

$$\prod_{i=1}^{L} \left[c_i \leq \sum_{j=1}^{M} T_{ij} \right] \left[\sum_{j=1}^{M} H_{ij} \leq c_i \right].$$

We can illustrate this property as follows. The assembly shown in Figure 2 has both properties 1 and 2. Based on the set of assembly sequences shown in Figure 3, we can write Type 2 precedence relations of the following form:

$$c_1 \leq c_2c_3c_4c_5 + c_2c_3c_4c_5 + c_3c_4c_5 + c_3c_5 + c_2c_4c_5 + c_2 + c_3c_5 + c_2 + c_2c_3c_4 + c_2,$$

and

$$true + true + c_2c_3 + c_2c_3c_4c_5 + c_2c_3 + c_2c_3c_4c_5 + c_2c_3c_4c_5 + c_2c_3c_4c_5 + c_5 + c_2c_3c_4c_5 \leq c_1.$$

By writing these relations for all $i = 1, \ldots, 5$, the resulting set of expressions can be simplified to obtain one non-redundant relation:

$$c_3 \leq c_1c_5 + c_2c_4.$$

This Type 2 precedence relation also specifies correctly and completely the set of feasible assembly sequences for the example in Figure 2. It corresponds to our intuitive interpretation of the precedence which requires that the connection between stick and receptacle be established before the ends are connected.

Properties 1 and 2 are the necessary and sufficient conditions for the simplification of Type 2 precedence relations. Property 1 asserts that shorter subsequences are not geometrically blocked by the omission of connections. Property 2 asserts that longer sequences are not made unstable by additional connections, and therefore that the feasibility of the resulting state is guaranteed.. Violation of Property 1 generally requires the occurrence of mechanisms to establish the interaction between connections. In the current experiments with precedence relations, we have assumed that no such mechanisms are present. Violation of Property 2 generally requires that unstable states be reachable through feasible operations. Figure 9 shows an example of an assembly which does not satisfy Property 2 when gravity is present. In most of the current experiments we have assumed no gravity and therefore stable subassemblies. Properties 1 and

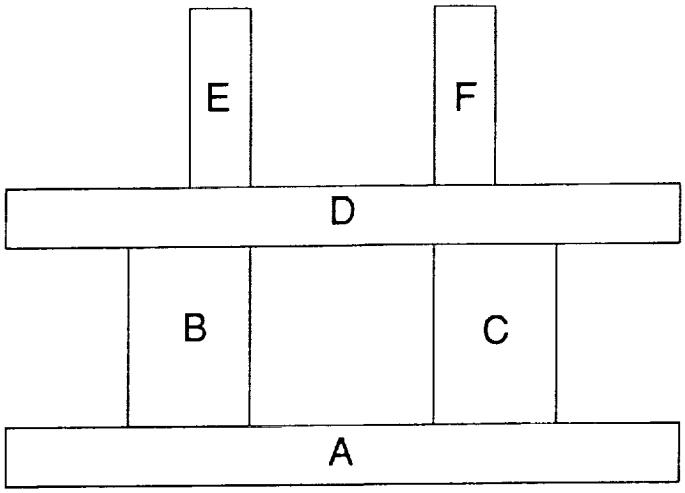

Figure 8 Example of an assembly which violates Property 2, but not Property 1. In this assembly, the configuration ABD is assumed stable, but ABDF is not stable. Therefore, we can execute the sequence of connections [A-B D-F] and the sequence [A-B B-D D-E D-F], but not the sequence [A-B B-D D-F]. This violates Property 2 and does not give correct Type 2 precedence relations of the simplified form. Note that this assembly has Property 1, and the state ABDF would be identified as infeasible in the generation of Type 1 precedence relations which are correct.

2 hold for the examples that we are treating in the current experiments, but as shown by the examples above, there are common cases where these properties may be violated,

Precedence relations provide an *implicit* representation of assembly sequences in the sense that decisions regarding the next step in the sequence may be made through local consideration of the current and previous states. On the other hand, an assembly sequence itself is an *explicit* representation of the assembly sequence and if a new state is entered, an entire sequence must be generated in order to guarantee its correctness. This distinction becomes important in the implementation of assembly sequence plans in real-time execution. In this mode of operation the choice of a next correct operation may be made without recourse to explicit planning of the entire sequence. On the other hand, using the types of precedence relations we have defined there is no guarantee that a next correct step is also going to be a step leading to a complete feasible sequence. Therefore, we have defined a real-time property of assembly sequence representations which requires that all feasible assembly subsequences which start at the initial state have a non-empty set of following subsequences which reach the goal. In reference [14] we have described algorithms which generate precedence relations which guarantee the real-time property to hold.

An example of an assembly in which the type 1 precedence relations do not provide the real-time property is shown in Figure 10. In this case the extended algorithms are required in order to generate a set of precedence relations guaranteeing the real time property.

5. Evaluation and Selection of Assembly Plans

The assembly planning algorithms described in previous sections generate the set of all feasible assembly sequences. The formal definition of these structures and algorithms which

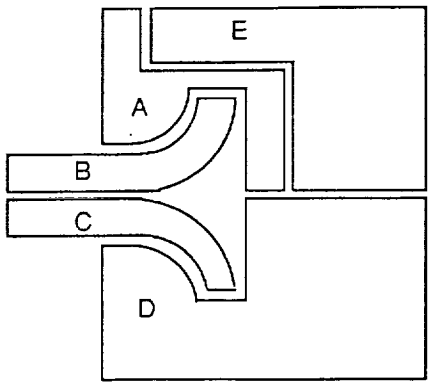

Figure 9 Example of an assembly which does not have the real-time property for Type 1 Precedence Relations. While B and C can be inserted individually at any time, if B-C are connected early in the sequence, several additional feasible steps may be taken, but do not lead to a feasible sequence. A deadend is reached when B-C trues to mate with A-E-D. The real-time precedence relations for this example are generated by our extended algorithm [14].

incorporate feasibility predicates is a necessary step toward the development of techniques which can be used to evaluate and select plans for particular applications. In this section we described several approaches to the evaluation and selection of plans.

In practice we would like to select a set of candidate assembly plans which most nearly meet our needs for a particular purpose. This selection requires the definition of evaluation measures and the implementation of search techniques in order to select appropriate plans. This evaluation and selection process would ideally occur in parallel with the generation of the plans themselves. The combinatorial explosion in the number of possible sequences makes it desirable to limit the search as early as possible and therefore to incorporate these evaluation measures as early as possible into the search. In this section we will summarize results for three different evaluation functions.

In practice, we would often like to choose assembly sequences which both minimize the complexity of the task execution as well as the complexity of the fixturing and manipulation to maintain the intermediate sub-assembly states. One possibility for such an evaluation function is a weighted combination of the complexity of the assembly tasks and the relative degrees of freedom of the parts in the intermediate sub-assemblies:

$$W_1(t) = \begin{cases} k_D \cdot D(n) \quad if \quad t = (n) \\ k_D \cdot D(n) + k_C \cdot C(h) + W_1(t_1) + W_1(t_2) \\ \qquad if\ t = (n, h, t_1, t_2), \end{cases}$$

where

D(n) = measure of the relative degrees of freedom of the parts of the subassembly,

TREE	COST
A-B-C	11
B-A-C	13
B-C-A	13
C-B-A	11
D-A-C	15
D-C-A	14
C-D-A	14
E-C-A	15
E-A-C	14
A-E-C	14

Table 1 Table of costs using the weighted evaluation function discussed in the text for the example in Figure 2. The preferred sequences were those which attached the cap or handle to the receptacle first, providing a stable subassembly for insertion of the stick.

$C(h)$ = measure of the complexity of the assembly task whose corresponding and-arc is h,

k_D = weight given to the relative degrees of freedom of the parts of the subassembly, and

k_C = weight given to task's complexity measure.

Sub-assemblies in which the number of relative degrees of freedom of the parts is high are more difficult to manipulate because there are fewer orientations in which they are stable and there are fewer options for grasping. A variety of factors can be included in a measure of complexity of assembly tasks: time duration, reliability, fixture requirements, cost of resources. As one example, we have established a ranking of assembly tasks based on threaded contacts, cylindrical contacts, and planer contacts. Either exhaustive or heuristic search techniques may be utilized to explore the alternative sequences relative to these evaluation functions. In this case, an evaluation function W_1 may be defined as an admissible heuristic function and incorporated into a heuristic search technique such as the AO* algorithm. Based on these search techniques, the resulting ranking of alternative sequences in terms of these evaluation functions are shown in Table 1.

Another possible metric to assess the quality of an assembly sequence is the number of distinct sequences in which the assembly tasks can be executed. For some applications it may be desirable to have one fixed set of assembly tasks all of which are executed. Instead of allowing all possibilities given by the AND/OR graph it may be preferred to allow only the possibilities given by one assembly tree. The assembly tree that allows the maximum number of distinct sequences is preferred because it gives more flexibility in the scheduling of tasks. Given an assembly tree, the number of distinct sequences in which the assembly task can be executed can be computed recursively. For this evaluation function an admissible heuristic can

also be found. The resulting evaluation of alternative assembly trees from Figure 2 suggests that sequences D-C-A/C-D-A and E–A–C/A-E–C from Figure 3 each are represented by the same solution tree for the assembly task while the other six assembly trees allow only one sequence.

A third possibility for a metric to assess the quality of an assembly tree is its depth. Assuming that the assembly work station operates in cycles during which one or more assembly tasks are executed, the depth of an assembly tree is given by the minimum number of cycles that are required to complete the assembly. Given an assembly tree, the depth metric yields an admissible heuristic function which can be implemented as an efficient search technique. For the example in Figure 2, the assembly trees found using metric 2 above also have depth two since it is possible to execute two assembly tasks simultaneously and therefore to complete the assembly in two cycles. Of course, simultaneity of assembly tasks requires the availability of resources and this metric would only be used if the required resources are available.

6. The PLEIDEAS System

While a fully automated assembly system is one goal of the techniques described here, many of these approaches will be utilized first in an interactive mode where the user is a critic for alternative assembly sequences generated automatically from CAD-based descriptions. The first generation of such an interactive assembly planning environment integrates a set of software modules for parts modeling, and planning. The system, which we call PLEIDEAS (PLanning Environment for Integrated Design of Assembly Systems), has now been used to demonstrate initial experiments in transferring a CAD-based parts description to a fully developed assembly plan.

Figure 11 illustrates the elements of the PLEIDEAS system. They include:

CATIA – CATIA is a solid modeling and mechanical CAD design system available commercially from IBM. CATIA uses both CSG and Boundary representations, and provides facilities for Boolean operations on solid models of the parts. Several parts may be built in the same model and viewed simultaneously as shown in Figure 12. However, there is no explicit representation for assembly relationships in CATIA, and relative parts positions must be described in order to represent assembly relationships. The partially assembled parts for the same example are shown in Figure 12. For our purposes, CATIA provides a convenient tool for generating and manipulating solid models of parts. It provides sufficient boundary model information to generate many assembly relationships, and provides good capabilities for visualization of parts and subassemblies. An additional advantage of the integration of this commercial system is the availability of existing designs which have been implemented using CATIA and the opportunity to interact more directly with practicing designers.

GEOS – GEOS [23,24] is a variational geometric modeling system under development at Rensselaer by Professor Joshua Turner. It is intended for modeling parts and assemblies and includes facilities to incorporate tolerance information. GEOS implements an object-oriented environment, including classes and methods with inheritance. GEOS model data is organized

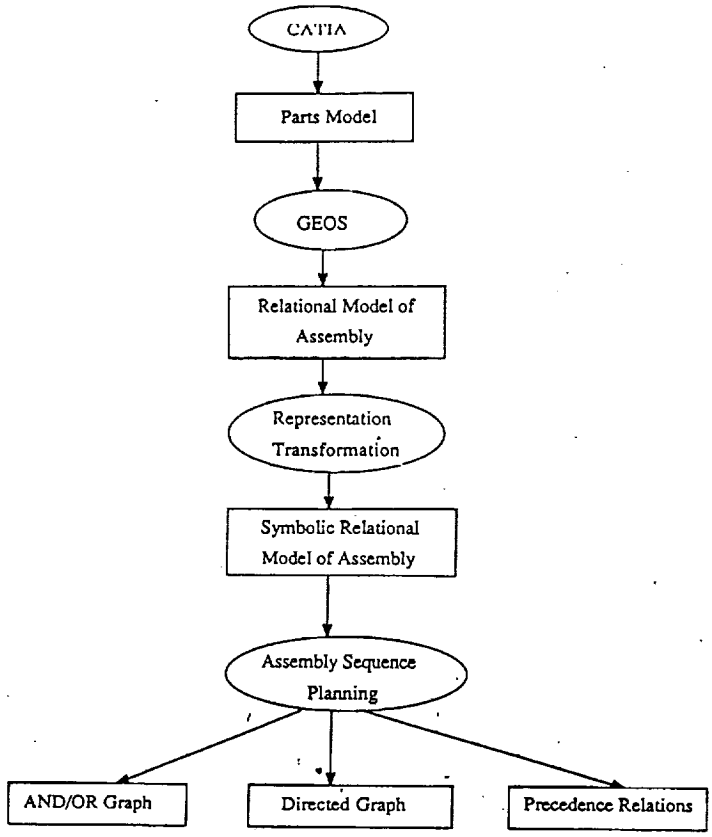

Figure 10 Overview of the PLEIDEAS environment for assembly sequence planning.

into data structures called objects or nodes. A GEOS display for the four-part example is shown in Figure 12b. Typical GEOS nodes are point nodes, line nodes, and surface nodes. GEOS nodes are grouped together into entities. Entities are the highest level of structure visible to the user. There are three types of entities:

1. Menu entity — contains all the menus used by the GEOS dialog manager,
2. Drawing entity — corresponds to a paper drawing and contains a collection of 2D components.
3. Space entity — corresponds to a three-dimensional world and contains a collection of 3D components.

In GEOS, the space entity can be used to model the assembly. Each space entity has a list of region nodes, and region nodes contain body nodes. The body nodes can be used to model the parts in the assembly. The two types of nodes used to model the geometric contacts and mechanical attachment relationships between the parts are those defined previously in our relational model (Section 2). A contact node models the contacts from the relational model

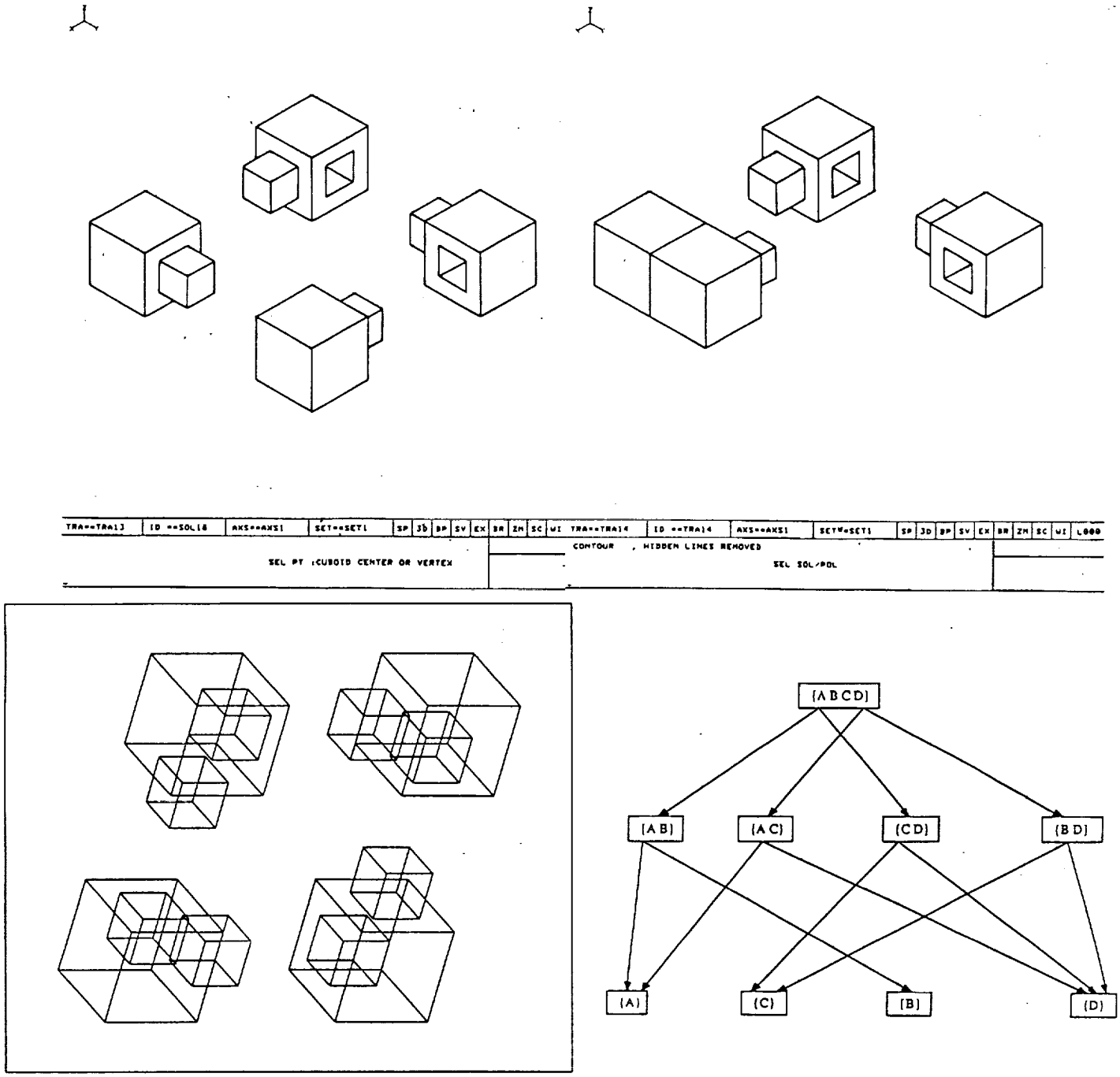

Figure 11 Example of the sequence of procedures in the PLEIDEAS system. (a). Output of the CATIA system showing the parts solid models. (b). GEOS display of object-oriented model of assembly relations, (c). AND/OR graph output of the planning procedure.

and stores the pointers to the two body nodes, and the pointers to the contacting surfaces. An attachment node includes the pointer to the agent nodes and the target nodes. In the current mode of operation, GEOS imports parts models from CATIA and converts them directly to GEOS representation. The assembly relationships are now input interactively.

Relational Model – The relational model which is used in the assembly planning software modules is obtained by data format conversion from the GEOS C-environment to a LISP-environment. The resulting symbolic relational model contains all of the relational and attribute information needed for assembly sequence planning, including evaluation of feasibility predicates for local geometric feasibility and attachments.

Assembly Sequence Planning – The assembly sequence planner utilizes the symbolic representation of the relational model to generate the AND/OR graph representation of all feasible assembly plans based on the evaluation of local geometric and attachment feasibility predicates. The AND/OR graph for the four-part example is shown in Figure 12c. The explicit assembly sequence lists, and the directed graph of assembly states can be generated from this representation.

Precedence Relations – Type 1 and Type 2 precedence relations are generated from the AND/OR graph representation using implementations of the algorithms described in [14]. For the four-part example in Figure 12, these are:

Type 1:

$$c_1 \rightarrow x_3 \cdot x_4 \qquad c_2 \rightarrow x_1 \cdot x_3 \qquad c_3 \rightarrow x_2 \cdot x_4 \qquad c_4 \rightarrow x_1 \cdot x_2,$$

Type 2:

$$c_1 \leq c_2 \cdot c_3 + c_4 \qquad c_2 \leq c_1 \cdot c_4 + c_3 \qquad c_3 \leq c_1 \cdot c_4 + c_2 \qquad c_4 \leq c_2 \cdot c_3 + c_1.$$

An additional example of the application of these planning tools is provided by the ball point pen shown in Figure 13. This is an example which was used by both Bourjault [20] and DeFazio and Whitney [21] in their work on interactive sequence planning. This example has 6 parts and 5 connections with a resulting 12 feasible sequences for assembly. The AND/OR graph has 17 nodes, and the directed graph has 24 nodes. The precedence relations which are generated for this example are:

Type 1:

$$c_4 \rightarrow x_1 \cdot x_2 \qquad c_3 \rightarrow x_4 \qquad c_1 \rightarrow x_5,$$

Type 2:

$$c_1 \leq c_5 \qquad c_3 \leq c_4 \qquad c_4 \leq c_2 + c_1$$

where the Type 1 precedence relations are identical to those obtained from the interactive methods in [21].

7. Conclusions

This paper summarizes work in progress on the development of a theoretical framework and implementation of assembly sequence planning tools. The principal focus has been the

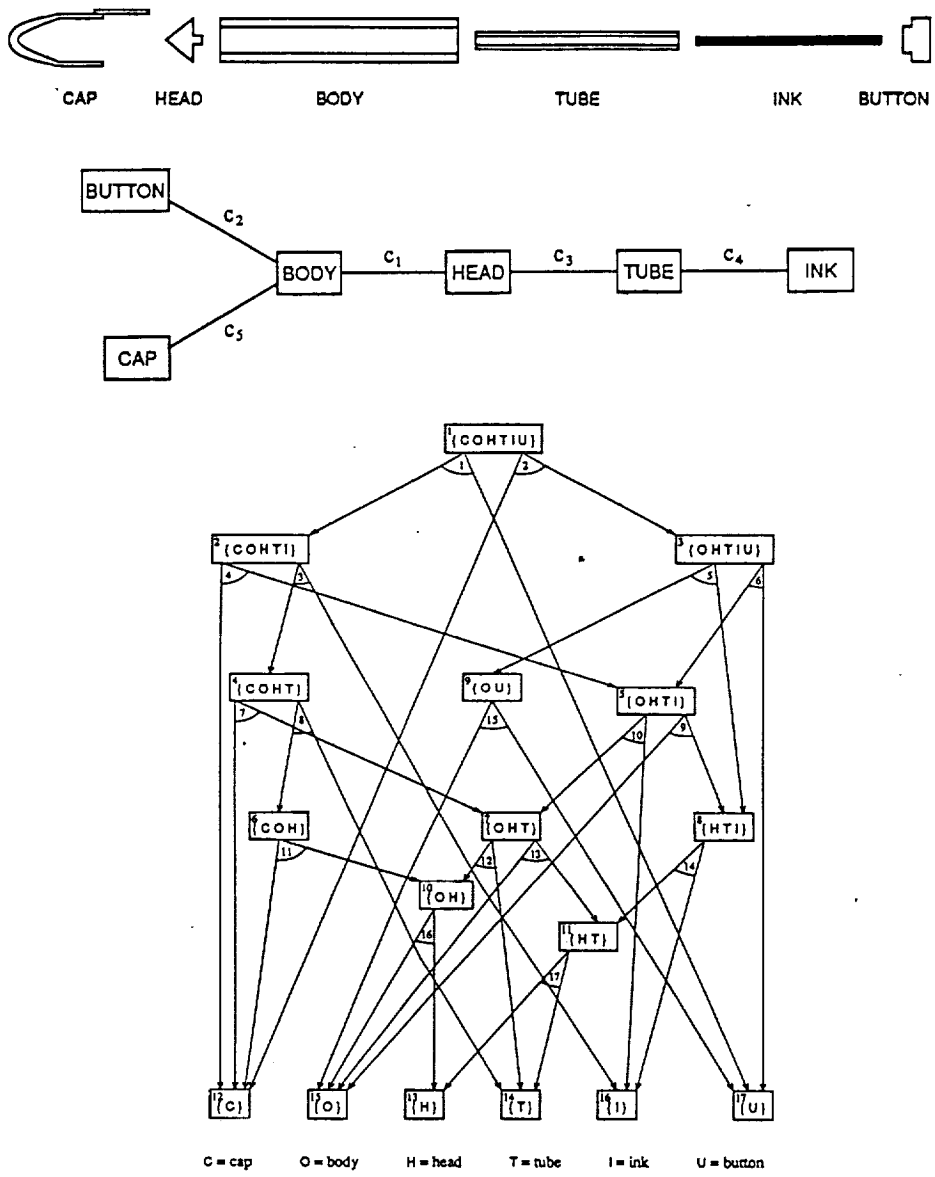

Figure 12 Ball-point pen example.

formalization of assembly sequence plan representations, the proof of equivalence of different representations, and the implementation of algorithms to generate and transform these plan representations. These representation tools and algorithms form the basis for implementation into a system of software modules which provides an environment for interactive design and evaluation of assembly system design alternatives.

There are a number of additions and extensions to the current system which are of both theoretical and practical importance:

1. *Feasibility predicates* — Within the framework which we have described, a large number of different resource independent and resource dependent feasibility predicates may be defined. For practical application, we need to incorporate a global geometric feasibility algorithm.

There are several candidates here, but many existing algorithms which plan explicit paths are too computationally intensive. An extended state stability criterion is also an important extension, but again will require additional research.

2. *Hierarchy* — Many assembly products are designed with natural hierarchies of subassemblies in order to maintain the stability of subunits resulting in both functional and manufacturing advantages. Such hierarchies fit naturally into the AND/OR graph framework and can be used to simplify the search procedure by enforcing subgoals.
3. *Uncertainty and Tolerances* — Assembly product designs have specified tolerances on the dimensions and shapes of parts. These tolerances are usually matched to the functional specification of the parts, and often reflect an implicit assumption by the designer about assembly sequencing. Incorporation of tolerance information into feasibility predicates for sequence planning is an important extension to our current work. We expect this analysis to lead to formal methods to incorporate other sources of uncertainty due to fixtures or manipulators.
4. *Mechanisms* — In our current analysis and experiments, we do not permit an assembly to occur in more than one final state. In practice, many assemblies are complex mechanical mechanisms whose function depends upon the occurrence of multiple states, and the assembly sequence may depend upon the ability to alter state during the assembly process. These considerations fall within the framework which we have defined, but we have not attempted to implement these considerations in detail.
5. *Efficiency* — The emphasis in this work has been in complete and correct representations of all feasible assembly sequences. In practice, the combinatorial search of all possible sequences is often prohibitive. We view this complete and correct specification as a necessary precursor to more efficient decomposition and search techniques which are implemented in a partial representation space. Our experiments with evaluation criteria discussed in Section 5 provide examples of heuristic search within the AND/OR graph space, without requiring that the entire graph be generated.
6. *Manipulation planning* — Implementation of an assembly system requires links between the sequence planner and the manipulation planner or task-level programmer. There are natural extensions of the structure described here which incorporate feasibility tests on resource dependent properties such as gripper geometry or force exerted. The detailed planning of manipulation and fine-motion actions is outside the direct scope of this work, but will be essential for automation of such systems. While each of these topics is a difficult research topic in itself, the abstracted representation of manipulation capability must be present for adequate planning of sequence based on resource dependent considerations.
7. *Parts design* — We have emphasized the planning of feasible sequences given a specification of parts and assembly relationships. Clearly parts redesign is one of the important links to successful manufacturing systems. A PLEIDEAS type of planning environment would permit the exploration of parts geometry and assembly modifications using the CATIA system, and an opportunity to evaluate the impact of design changes on assembly plans.

Acknowledgements

This work has been supported by the New York State Center for Advanced Technology in Automation and Robotics and the NASA Center for Intelligent Robotics Systems for Space Exploration at Rensselaer Polytechnic Institute. Additional support has been received from the Conselho Nacional de Desenvolvimento Cientifico e Tecnologico (Brazil), The Jet Propulsion Laboratory of the California Institute of Technology, and The Robotics Institute of Carnegie Mellon University.

REFERENCES

1. Fikes, R., and N. Nilsson, "STRIPS: A New Application of Theorem Proving to Problem Solving," *Artificial Intelligence* 2, pp. 189–208, 1972.
2. Sacerdoti, E.D., "Planning in a Hierarchy of Abstraction Spaces," *Third International Joint Conference on Artificial Intelligence*, pp. 412–422, Stanford Research Institute Publications, 1973.
3. Wilkins, D., "Domain-independent Planning: Representation and Plan Generation," *Artificial Intelligence* 22 PP. 269–301, 1984.
4. Chapman, D., "Planning for Conjunctive Goals," *Artificial Intelligence* 32 pp. 333–377, 1987.
5. Korf, R.E., " Planning as Search: A Quantitative Approach", *Artificial Intelligence* 33 pp. 65–88, 1987.
6. Homem de Mello, L. S., and A. C. Sanderson, "AND/OR Graph Representation of Assembly Plans," *Proc. of the Fifth National Conference on Artificial Intelligence AAAI-86* pp. 1113–1119, Morgan Kaufman, 1986.
7. Homem de Mello, L. S., and A. C. Sanderson, "Task Planning and Control Synthesis for Flexible Assembly Systems," in *Machine Intelligence and Knowledge Engineering for Robotic Applications* NATO ASI Series, Vol. F33, Ed. A.K.C.Wong and A.Pugh, Springer-Verlag, Berlin, 1987.
8. Homem de Mello, L. S., and A. C. Sanderson, "Automatic Generation of Mechanical Assembly Sequences," Technical Report CMU-RI-TR-88–19, The Robotics Institute, Carnegie Mellon University, December, 1988.
9. Homem de Mello, L. S., and A. C. Sanderson, "Precedence Relationship Representations of Mechanical Assembly Sequences," in *Proc. 2nd NASA Workshop on Space Telerobotics* Pasadena, CA, January, 1989.
10. Sanderson, A. C., and L. S. Homem de Mello, "Automatic generation of mechanical assembly sequences,". In J. Turner, M. Wozny, and K. Preiss, eds, *Proc. 1988 IFIP/NSF Workshop on Geometric Modeling*. Elsevier, 1989.
11. Homem de Mello, L. S., and A. C. Sanderson, "Representations of assembly sequences," In. *Proc. 11th International Joint Conference on Artificial Intelligence*, Morgan Kaufman, 1989.
12. Homem de Mello, L. S., and A. C. Sanderson, "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences," *IEEE Transactions on Robotics and Automation*, in press, 1989.

13. Homem de Mello, L. S., Task Sequence Planning for Robotic Assembly, PhD Dissertation, Electrical and Computer Engineering Department, Carnegie Mellon University, May, 1989.
14. H. Zhang, " Generation of Precedence Relations for Mechanical Assemblies," Master's Thesis, Rensselaer Polytechnic Institute, August, 1989.
15. L. I. Lieberman and M. A. Wesley, "AUTOPASS: An automatic programming system for computer controlled mechanical assembly," *IBM Journal of Research and Development,* 21(4):321–333, July, 1977.
16. C. M. Eastman, "The design of assemblies," *SAE Technical Paper Series,* Society of Automotive Engineers, 1981.
17. K. Lee and D. C. Gossard, "A hierarchical data structure for representing assemblies: Part 1.," *CAD* 17(1):15–19, Jan/Feb, 1985.
18. Lozano-Perez, T., and M. Wesley, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Comm. ACM* 22 pp. 560–570, 1979.
19. Bonenschanscher, N., "Subassembly stability," In *Proc. of AAAI-88*, pp 780–785, Morgan Kaufman, August, 1988.
20. Bourjault, A., Contribution a une Approche Methodologique de L'Assemblage Automatise: Elaboration Automatique des Sequences Operatoires, These d'Etat, Universite de Franche-Comte, Besancon, France, November, 1984.
21. DeFazio, T. L., and D. E. Whitney, "Simplified Generation of All Mechanical Assembly Sequences", *IEEE Journal of Robotics and Automation* RA-3(6), pp. 640–658, December, 1987. See Corrections, same journal RA-4(6), pp. 705–708, December, 1988.
22. M. M. Lui, "Generation and evaluation of mechanical assembly sequences using the liaison-sequence method," Master's Thesis, MIT, May, 1988.
23. J. Turner, "GEOS design notes," Rensselaer Design Research Center, Rensselaer Polytechnic Institute, February, 1989.
24. J. Turner, "GEOS user notes," Rensselaer Design Research Center, Rensselaer Polytechnic Institute, February, 1989.